Corel® Paradox® 8

March Newsletter **March 1, 1998**

Corel® Paradox® 8 Add-Ins

This month's newsletter shows you how to transform your existing Delphi™ form into a Corel® Paradox® 8 add-in. The add-in allows you to write a program in Delphi and integrate it into Paradox, enabling communication.

To help with this, a set of classes has been provided that allows the program to share different parts of Corel Paradox 8, like menus and the desktop and its different services. Also, an object type for using add-ins directly from an ObjectPAL™ program is included.

The main advantage of the Corel Paradox 8 add-in is the power of the Delphi language that comes with it. Also, making this an add-in from an existing Delphi form takes only a few minutes. Once the form has been added, your users will never notice the difference between your Delphi form and a regular Paradox form!

- **Transforming a Delphi Form into a Corel Paradox 8 Add-In**
- **Creating Menu Items and Merging them into Corel Paradox 8**
- **Registering your Corel Paradox 8 Add-In**
- **Tips & Tricks**

Transforming a Delphi™ Form into a Corel® Paradox® 8 Add-In

Corel® Paradox® 8 includes all the necessary classes in the form of a Delphi™ software developer kit (SDK). If you accepted the default configuration upon setup, the SDK is located in the folder **C:\Corel\Suite8\Paradox\Addins\Sdk.** If you performed a customer installation and you do not have the SDK file, you must run the Corel Paradox 8 setup to install all the necessary files.

5. First, your add-in must be part of a DLL. If it is not, you must create a new DLL project. If your add-in is part of a DLL, skip to step 3.

6. To create a new DLL project, select **File/New …**, select the **New** tab, then double-click the DLL icon. Next, copy your existing Delphi form into our new project directory and add it to your project file (**Project/Add to project**). If you do not have a form create a new one. (**File/New…, New** tab, double-click the Form icon.)

7. Two units are required to make your form add-in compliant: **PdxAddIn** and **PdxForms**. Put them into the **Uses** clause of your project (Library) file and in your form (Unit) file. **PdxForms** contains all the necessary definitions and code for making Corel Paradox 8 recognize your form. The **PdxAddIn** will be discussed later.

8. For these two libraries to work, you need to put two paths in your libraries path:
   - **add-ins SDK path**
   - **Delphi 2 library path** (needed only if you are using Delphi 3).

For example, the files may be located in **C:\Corel\Suite8\Paradox\Addins\Sdk** and **C:\Borland\Delphi 3\Lib\Delphi2**. From the menu, select **Tools/Environment Options…**, select the **Library** tab, where you will see the Library path box.

5. In your project (`Library`) file, there is a begin…end section. Add the following code to make your add-in publicly available:

```
Paradox.RegisterFormClass( 'Example', TExample );
```

6. In the source code of your form, you will find this line:

```
type
  TForm1 = class(TForm)
```

change it to

```
type
 TForm1 = class(TParadoxForm)
```

The Delphi form has now been transformed into a Corel Paradox 8 add-in.

Creating Menu Items and Merging them into Corel® Paradox® 8

The simplest way to create a menu item and merge it into Corel® Paradox® 8 is to use a special (invisible) form called Virtual Paradox Desktop. This form is also included in the add-in SDK and must be added to your project. Copy the files **PdxDeskW.*** from the SDK directory to your project directory. Add them to your project. You can open this form by selecting **View/Forms...** and then clicking **f_ParadoxDesktop**.

To add a menu item:

1. Open the form.
2. Right-click on a menu item (e.g. **Tools** in our example) and select the Menu Designer.
3. Create an item (type in the name **Addin** and the caption **&Addin**).
4. Select the **OnClick** function and input the following code:
   (Note that in the code, TExample is the type of our example Delphi form.)

```
Example :=
TExample.Create(Application);
Example.ShowModal;
Example.Free;
```

5. You also need **AnAddin** in the **USES** clause.
6. Add this line in the **VAR** clause of your modified **PdxDeskW.pas**. This will instantiate the TExample Delphi form.

```
Example: TExample;
```

Corel Paradox 8 will automatically know about the menus when you register your DLL.

Registering your Corel® Paradox® 8 Add-In

DLLs must be registered in order to be recognized by Corel® Paradox® 8. To do this:

1. Start Corel Paradox 8.
2. Select **Tools/Register Add-In** and click **Add**.
3. Select your DLL file and choose **OK**.
4. Exit and restart Corel Paradox 8.

Once the registration is complete, the recently created menu item(s) will be visible from within the Corel Paradox 8 menu hierarchy.

Corel® Paradox® 8 Add-In Tips & Tricks

- Paradox Global Variable
- Add-in Form OPAL Type
- Making Delphi™ Properties Available Within Corel® Paradox® 8
- Sending Windows Messages

Paradox Global Variable

Using the **PdxAddIn** can assist the communication between your new add-in and Corel® Paradox® 8. The **PdxAddIn** contains a global variable called Paradox, of type TParadoxAddin

The Paradox global variable has many available members:

Paradox

RegisterMenu
RegisterForm
RegisterFormClass
ClearMenus;
ClearForms;
ResetStatusRegions
SetStatusRegions
SetStatusText
Property ParadoxInterface:IParadox
Property DesktopHandle:HWND
Property DesktopClientHandle:HWND

Paradox.ParadoxInterface

GetDesktopHandle:HWND
GetDesktopClientHandle:HWND
SetMenu(hwnd:HWND; menu:HMENU)
SetDeskMenu
RevertMenu
RebuildMenu
SetStatusRegions(nRegions:Integer; RegionSizes:PInteger)
SetStatusText(nRegion:Integer; RegionText:PChar)
ShowHelp(HelpContext:Integer)
DoMenuAction(Command:Word)
RaiseError(ErrorId:Integer; ErrorText:PChar)
GetResourceHandle:Longint
GetAutomationInterface:IpdoxAutoPascal

Add-in Form OPAL Type

From within ObjectPAL™, we can communicate with an add-in using an instance of the **AddinForm** type. Here is an overview of its methods:

Attach
BringToTop
Close
CloseQuery
EnumForms
GetPosition
GetPropertyAsInteger
GetPropertyAsNumber
GetPropertyAsString
GetTitle
Hide
IsAssigned
IsMaximized
isMinimized
IsVisible
maximize
menuAction
minimize
open
postMessage
sendMessage
setPosition
setProperty
setTitle
show
wait
windowHandle

To communicate with your add-in, you must create an instance of the **AddinForm** type from within a Corel Paradox 8 form.

For example:

```
var
   Addin    AddinForm
endVar
[...]
;first check if the add-in is already
opened.
;Important Note: Here Example Title is
the caption of the Delphi form
;window and Example is its registered
name
if NOT Addin.attach("Example Title")
then
    Addin.open("Example")
endIf
Addin.show()
```

For more information about **AddinForm**, consult ObjectPAL help.

Making Delphi Properties Available Within Corel Paradox 8

Those who are familiar with Delphi will be familiar with Properties. A Property is like a publicly available member variable, public enough to be read from an ObjectPAL program.

In our example, we use a Property called **Status**, it is an **Integer**. The Delphi code for this is as follows:

```
private
  FStatus: Integer;
Published
  property Status: Integer read FStatus
write Fstatus;
```

From an **AddinForm** object, you can get its value by calling:

```
Addin.GetPropertyAsInteger("Status")
```

You can also set it by writing:

```
Addin.SetProperty("Status",3)
```

Sending Windows Messages

Window handles have many uses, including sending Windows messages.

From ObjectPAL's **AddinForm** object, the window handle of your Delphi form is a **LongInt** returned by the function `windowHandle()`.

From Delphi's **Paradox** object, the method `GetDesktopHandle()` returns the handle of the main Paradox window, and `GetDesktopClientHandle()` returns the handle of its MDI client area. Now that you have the handles, you can use `SendMessage()` to get any kind of information.